METHOD AND SYSTEM FOR PROVIDING ELECTRONIC COMMERCE ACTIONS BASED ON SEMANTICALLY LABELED STRINGS

Reference to Related Applications

This is a continuation-in-part of U.S. Patent Application Serial No. 09/588,411, entitled "METHOD AND SYSTEM FOR SEMANTICALLY LABELING STRINGS AND PROVIDING ACTIONS BASED ON SEMANTICALLY LABELED STRINGS", filed June 6, 2000, which is incorporated by reference herein.

Technical Field

This invention relates to a method and system for providing electronic commerce actions that may be performed based on semantically labeled strings.

Background of the Invention

Electronic documents typically include semantic information that would be helpful if the information was recognized as such. Recognition and use of this semantic information could result in increased interoperability between desktop software applications and other desktop applications and/or web-based applications. Recognition of this semantic information may also provide benefits in electronic commerce.

Electronic commerce is increasing in popularity as users become accustomed to transacting business over the Internet. A fundamental way for a retailer to generate traffic to its website is via a hyperlink. Although hyperlinks are useful, they do have some drawbacks. One drawback of hyperlinks is that they are static. In other words, a

hyperlink navigates a user's web browser to a single webpage without regard for the user's interests. The user is expected to decide to which webpage to navigate to and it is a one-to-one navigation. Moreover, hyperlinks are typically not automatically applied to document content (although some programs do recognize them as hyperlinks after they are entered by a user).

Thus, there is a need for a dynamic e-commerce tool that may be automatically applied to documents that a user creates, receives or views.

Summary of the Invention

The present invention provides a method and system for recognizing strings, labeling the strings with a semantic category and providing e-commerce actions based on the category. The semantic category may include a type label and other metadata. Recognizer plug-ins perform the recognition of particular strings in an electronic document. The recognizer plug-ins may be packaged with an application program module or they may be written by third parties to recognize particular strings that are of interest. Action plug-ins provide possible actions to be presented to the user based upon the type label associated with the string. Tradenames, trademarks, formal names or types of consumer products may be labeled and actions to buy the products may be presented. The metadata may be used to implement coupon and affiliate programs to reward frequent shoppers or frequent recommenders. Numerous other e-commerce opportunities are presented via the semantic category and the metadata.

These and other features, advantages, and aspects of the present invention may be more clearly understood and appreciated from a review of the following detailed description of the disclosed embodiments and by reference to the appended drawings and claims.

Brief Description of the Drawings

Fig. 1 is a block diagram of a computer that provides the exemplary operating environment for the present invention.

Fig. 2 is a block diagram illustrating an exemplary architecture for use in

conjunction with an embodiment of the present invention.

Fig. 3 is a flow chart illustrating a method for semantically labeling strings during creation of an electronic document.

Fig. 4 is an illustration of a display of a semantic category and its associated dropdown menu.

Detailed Description

The present invention is directed toward a method and system for semantically labeling strings of text and providing a selection of electronic commerce actions that may be performed based on the semantically labeled strings. A string is defined as a data structure composed of a sequence of characters usually representing human-readable text.

In one embodiment, the invention is incorporated into a suite of application programs referred to as "OFFICE", and more particularly is incorporated into a preferred word processing application program entitled "WORD 10.0", a preferred spreadsheet application program entitled "EXCEL 10.0", a preferred e-mail application program entitled "OUTLOOK 10.0" and a preferred web browser application program entitled "INTERNET EXPLORER 6", all marketed by Microsoft Corporation of Redmond, Washington. Briefly described, the preferred application programs allow a user to create and edit electronic documents by entering characters, symbols, graphical objects, and commands.

Strings are recognized and annotated, or labeled, with a type label. After the strings are annotated with a type label, application program modules may use the type label and other metadata to provide users with a choice of electronic commerce actions. If the user's computer does not have any actions associated with that type label, the user may be provided with the option to surf to a download Uniform Resource Locator (URL) and download action plug-ins for that type label.

Having briefly described an embodiment of the present invention, an exemplary operating environment for the present invention is described below.

Exemplary Operating Environment

Referring now to the drawings, in which like numerals represent like elements throughout the several figures, aspects of the present invention and the exemplary operating environment will be described.

Fig. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of an application program that runs on an operating system in conjunction with a personal computer, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, cell phones, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to Fig. 1, an exemplary system for implementing the invention includes a conventional personal computer **20**, including a processing unit **21**, a system memory **22**, and a system bus **23** that couples the system memory to the processing unit **21**. The system memory **22** includes read only memory (ROM) **24** and random access memory (RAM) **25**. A basic input/output system **26** (BIOS), containing the basic routines that help to transfer information between elements within the personal computer **20**, such as during start-up, is stored in ROM **24**. The personal computer **20** further includes a hard disk drive **27**, a magnetic disk drive **28**, e.g., to read from or write to a removable disk **29**, and an optical disk drive **30**, e.g., for reading a CD-ROM disk **31** or to read from or write to other optical media. The hard disk drive **27**, magnetic disk drive

28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage for the personal computer 20. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD-ROM disk, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored in the drives and RAM 25, including an operating system 35, one or more application programs 36, a word processor program module 37 (or other type of program module), program data 38, and other program modules (not shown).

A user may enter commands and information into the personal computer 20 through a keyboard 40 and pointing device, such as a mouse 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers or printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be a server, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the personal computer 20, although only a memory storage device 50 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 51 and a

wide area network (WAN) **52**. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the personal computer **20** is connected to the LAN **51** through a network interface **53**. When used in a WAN networking environment, the personal computer **20** typically includes a modem **54** or other means for establishing communications over the WAN **52**, such as the Internet. The modem **54**, which may be internal or external, is connected to the system bus **23** via the serial port interface **46**. In a networked environment, program modules depicted relative to the personal computer **20**, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Fig. 2 is a block diagram illustrating an exemplary architecture **200** for use in conjunction with an embodiment of the present invention. The architecture includes an application program module **205**, such as word processor program module **37** (Fig. 1). The application program module **205** is able to communicate with a recognizer dynamic-link library **210** (hereinafter recognizer DLL) and an action dynamic-link library **215** (hereinafter action DLL) as a user is creating or editing an electronic document. The recognizer DLL **210** controls a number of recognizer plug-ins **220**. The action DLL **215** controls a number of action plug-ins **225**. The action DLL also controls a type-action database **230**.

In a preferred embodiment, the action plug-ins and recognizer plug-ins are Automation Servers. Automation Servers are well-known software components which are assembled into programs or add functionality to existing programs running on the Microsoft WINDOWS® operating system. Automation Servers may be written in a variety of computing languages and may be un-plugged from a program at run time without having to recompile the program. It should also be understood that, in a preferred embodiment, the action DLL and recognizer DLL are merged into a single

DLL.

The recognizer DLL **210** handles the distribution of strings from the electronic document running on the application program module **205** to the individual recognizer plug-ins **220**. The recognizer plug-ins **220** recognize particular strings in an electronic document, such as a word processing document, a spreadsheet document, a web page, etc. The recognizer plug-ins **220** may be packaged with the application program module **205** or they may be written by third parties to recognize particular strings that are of interest. Typically, the recognizer DLL **210** passes strings to the recognizer plug-ins **220** in one paragraph or cell value increments.

As part of recognizing certain strings as including semantic information, the recognizer plug-ins **220** determine which strings are to be labeled and how they are to be labeled. After receiving these results from the various recognizer plug-ins **220**, the recognizer DLL **210** sends semantic categories to the application program module. In a preferred embodiment, a semantic category comprises the recognized string, a type label, and a download URL. A semantic category may also comprise metadata. The recognizer plug-ins **220** each run separately and the recognizer DLL **210** is responsible for handling the asynchronicity that results from different recognizer plug-ins returning results with different delays.

After a string is labeled by a recognizer plug-in **220** and a semantic category is sent to the application program module **205**, the user of the application program module **205** will be able to execute actions that are associated with the type label of the semantic category. The action DLL **215** manages the action plug-ins **225** that are run to execute the actions. As with the recognizer plug-ins **220**, the action plug-ins **225** may be packaged with the application program module **205** or written by third parties to perform particular actions that are of interest to the third party. The action plug-ins provide possible actions to be presented to the user based upon the type label associated with the string. The action DLL **215** determines what type label the semantic category includes and cross-references the type label in the type-action database **230** with a list of actions to

determine what actions to present to the user. It should be understood that, in a preferred embodiment, the type-action database is not used. Instead, the list of actions is dynamically generated for each type by looking in the registry to determine which actions are installed and then querying the action DLLs to determine which types they apply to.

After the user chooses an action, the action DLL **215** manages the appropriate action plug-ins **225** and passes the necessary information between the action plug-ins and the application program module **205** so that the action plug-in may execute the desired action. Typically, the application program module sends the action DLL an automation request to invoke the action the user has selected.

As described above, the combination of the recognized string, type label, metadata and download URL is referred to herein as a semantic category. The type label is a semantic information label. The semantic category may also comprise metadata, which are hidden properties of the semantic category. An example of a semantic category may clarify the definition. Suppose a user enters the text "Gone With the Wind" into an electronic document. The string "Gone With the Wind" may be identified as a semantic category of type label "Book Title" and of type label "Movie Title". In addition, metadata such as the ISBN number may be returned by the recognizer plug-in to the application program module as part of the semantic category. A download URL may be provided with the type labels "Book Title" and "Movie Title" in case the user's machine has not stored action plug-ins for these type labels. For example, an action for the type label "Book Title" may be "Buy this Book" from an online retailer. If the user does not have the action plug-in DLL **225** corresponding to "Buy this book", then the download URL may be used to navigate the user's web browser to an appropriate website to download this action plug-in. In other implementations of the invention, multiple download URLs may be provided for a single type label.

It should also be understood that the present invention, in a preferred embodiment, also recognizes sequences of capitalized words that contain function words, and which are likely to be special, but for which there is no type label information. These

strings are typically labeled by a grammar checker program module.

The actions provided for a semantic category may utilize both the type label and the text of the recognized string. For example, a word processor program module may use a grammar checker as a recognizer plug-in to label strings that are person names. After a string has been labeled as a person's name, the word processor program module may, through a standard user interface mechanism, allow users to execute pertinent actions, such as looking up the person's name in the contacts folder in a personal information manager program module, sending electronic mail, or searching for the person's name in an HR database.

Having described an exemplary architecture, an exemplary method **300** for semantically labeling strings during document creation will be described below in reference to Figs. 2 and 3.

Method for Semantically Labeling Strings During Document Creation

Fig. 3 is a flow chart illustrating a method **300** for semantically labeling strings during creation of an electronic document. Those skilled in the art will appreciate that this is a computer-implemented process that is carried out by the computer in response to input from the user and instructions provided by a program module.

Referring to Fig. 3, the method **300** begins at start step **305** and proceeds to step **310** when a user opens an electronic document in application program module **205**. In a preferred embodiment, the electronic document is a word processing document or a spreadsheet document. However, the invention is not limited to either of these specific types of electronic documents.

At step **310**, the application program module **205** receives a new string, such as when the user enters a new paragraph into the electronic document or edits a previously entered paragraph. The method **300** then proceeds to step **315**.

At step **315**, the paragraph containing the new string is passed from the application program module **205** to the recognizer DLL **210**. The recognizer DLL is responsible for communicating with the application program module, managing the jobs

that need to be performed by the recognizer plug-ins, receiving results from the recognizer plug-ins and sending semantic category information to the application program module. At boot time, the recognizer DLL communicates with its recognizer plug-ins to determine what languages it supports, what types it can apply, etc. It should be understood that, in a preferred embodiment, a paragraph is passed to the recognizer DLL at step **315**. However, in alternative embodiments, a sentence, the contents of a spreadsheet cell, a section of the document, the entire document, etc. may be passed to the recognizer DLL. In other words, the present invention is not limited to simply passing a paragraph to the recognizer DLL. The method **300** then proceeds to step **320**.

Still referring to step **315**, the application program module **205** typically sends one paragraph at a time to the recognizer DLL. In addition, in a preferred embodiment, a grammar checker program module sends all semantic categories (without type labels) to the recognizer DLL that have been identified by the grammar checker program module. Passing these semantic categories (without type labels) to the recognizer DLL is important because doing so saves each recognizer plug-in from needing to decide whether something is a capitalized string interspersed with function words (a task that would require writing a number of regular expressions: Cap Cap Unc Cap; Cap Unc Cap; etc.). If a label is applied by a recognizer plug-in to a string the grammar checker program module labeled, the grammar checker label will then be removed.

At step **320**, during idle time, the paragraph (and information from the grammar checker program module) is passed to the recognizer plug-ins. The method then proceeds to step **325**.

It should be understood that, in a preferred embodiment, the recognizer DLL **210** maintains a job queue. If before the recognizer DLL **210** sends the paragraph to the recognizer plug-ins **220** the user edits the paragraph, then the job containing the edited paragraph is deleted and is not sent to the recognizer plug-ins. Then, a new job enters the queue at step **315** after the edited paragraph is received at step **310**. This job deletion is necessary to prevent the recognizer plug-ins from performing unnecessary work on a

paragraph that has been edited.

At step **325**, the recognizer plug-ins are executed on the paragraph to recognize keywords or perform other actions defined by the recognizer plug-in. As part of executing the recognizer plug-in, the paragraph may be broken into sentences by the recognizer plug-in. However, each recognizer plug-in is responsible for its own sentence-breaking. After the keywords are found at step **325**, then the method proceeds to step **330**.

At step **330**, the results from each of the recognizer plug-ins are received by the recognizer DLL. The method then proceeds to decision step **335**.

At decision step **335**, it is determined whether the paragraph that has been reviewed by the recognizer plug-ins has been edited after the paragraph was sent to the recognizer DLL. If so, then the method **300** returns to step **315** and the edited paragraph is received by the recognizer DLL from the application program module. If not, then the method proceeds to step **340**.

At step **340**, the results from the recognizer plug-ins are compiled into semantic categories by the recognizer DLL and the semantic categories are sent to the application program module. At step **345**, the application program module displays the semantic categories to the user in the electronic document. The method **300** then ends at step **399**.

As should be understood from the above description, the architecture for recognizing semantic categories permits third parties to develop recognizer plug-ins to identify strings of one or more particular types. The recognizer plug-ins communicate with the application program module and receive a string from the application program module. The recognizer plug-ins may apply recognition algorithms to the string and communicate the identity of recognized strings back to the application program module.

After a string is labeled with a particular type label, the user will be able to execute action plug-ins that pertain to that type label. The action plug-ins preferably are COM objects that are executed via communication between the application program module and the action DLL. Parameters necessary to execute the action (the HTML of

the string labeled as being of a particular type, the HTML of the string representing the current selection) will be passed from the application program module to the action DLL and, in turn, passed to the action plug-in.

Actions Assigned to Type Labels

An architecture for identifying and executing a set of actions associated with a semantic category may also be provided. This architecture comprises actions that apply to a particular type label (e.g. an action for book titles may be "Buy this book from shop.Microsoft.com") and executing those actions when the user so desires. An action is a user-initiated function applied to a typed string. For example, adding a name to the contacts folder is one action possible for a type label "Person name".

There is power and flexibility that results from allowing third party vendors, such as IT professionals, to design and write recognizer plug-ins and action plug-ins for deployment within an organization or for deployment on the World Wide Web. Some example actions that may be executed include:

Schedule a meeting

Create task

Display calendar

Add to contacts folder

Look up in contacts folder, address book, Windows Address Book (WAB), Global Address List (GAL), etc.

Insert address into document

Send mail to

Display EXPEDIA map

Stock quote lookup

Send instant message to

Different actions may be assigned to different type labels and these type label-action assignments may be stored in the type-action database **230**. Table 1 below illustrates some possible type label-action pairings.

Type Labels	Actions
Person name	Show contact info Add to contacts E-mail Insert address into document Send instant message to
Date	Show calendar for that day New task with that due date Schedule meeting that day
Place	Display EXPEDIA map Add to contacts
Address	Add to contacts
Phone number	Add to contacts
E-mail	Add to contacts
Date	Schedule a meeting
Task	Schedule a task
Meeting	Schedule a meeting

Table 1

For each type label, the type-action database **230** may store a download URL specified by the creator of the type label that users who do not have action-plug-ins or recognizer plug-ins for that semantic category type can go to in order to get action plug-ins and/or recognizer plug-ins. For example, the download URL for the type label "Book Title" might be microsoft.com/semanticcategories.asp. Once at that web page, a user may be offered downloads of various action plug-ins and recognizer plug-ins. There may also be an option on the user interface to navigate to the download URL so that recipients of documents with semantic categories can easily get the action plug-ins for those

semantic categories.

Storing Semantic Categories

Semantic categories may be stored as part of the electronic document along with other document information and may be available when a document is transmitted from one computer to another computer. In a preferred embodiment, storing semantic categories in an electronic document is controlled by an "Embed semantic categories" checkbox. The checkbox is on by default. Turning it off will prevent semantic categories in the document from being saved. The state of the checkbox is per document. The same checkbox controls saving for both .htm and .doc documents.

Checking a "Save semantic categories as XML properties" checkbox (off by default) will write out the text of all of the semantic categories in the document and their labels in the header of the HTML file in XML (that is using the same tags as are used inline, but surrounded by <xml> And </xml>) for easy identification and parsing by search engines and knowledge management systems.

Semantic categories may be saved as a unique namespace plus a tag name. A namespace is an XML construct for uniquely identifying a group of XML tags that belong to a logical category. Thus, every semantic category is uniquely identified by its nametag (e.g., "streetname") in addition to its namespace (e.g., "schemas-microsoft-com:outlook:contact")

Although the method **300** described above is one method for identifying semantic categories, there may be other mechanisms for identifying semantic categories. One mechanism is a grammar checker program module (not shown) connected to word processor program module **37**. Another mechanism is receiving a semantic category from another electronic document. For example, when text containing a semantic category is copied from one electronic document and pasted into another electronic document of the word processor program module **37**, the information identifying the semantic category is preserved and copied along with the copied text.

Displaying Semantic categories to the User

Referring now to Fig. 4, an illustration of a display of a semantic category **400** and its associated dropdown menu **405** will be described. It should be understood that Fig. 4 is an illustration of a semantic category **400** and dropdown menu **405** as displayed to a user by the application program module **205**.

The string **410** associated with semantic category **400** is the string "Bob Smith". As shown in Fig. 4, the string **410** of a semantic category **400** may be identified to the user by brackets **415**. Of course, many other devices such as coloring, underlining, icons, etc. may be used to indicate to the user that a particular string is a semantic category.

In a preferred embodiment, when the user hovers a cursor over the string **410** or places the insertion point within string **410**, then dropdown menu **405** is displayed to the user. The dropdown menu may display a list of actions associated with a semantic category. The dropdown menu may appear above and to the left of the semantic category string.

Typically, the first line of the dropdown menu indicates which string is the semantic category string (Bob Smith in Fig. 4) and what type the semantic category is (Person name in Fig. 4). Listed below the first line are actions **420** available for the semantic category type, such as "Send mail to...", "Insert Adress", and "Display contact information...".

The first item on the drop down menu below the separator line is "Check for new actions..." **425**. "Check for new actions..." **425** will appear only for semantic categories whose download URL is available to the application program module. If selected, "Check for new actions..." **425** uses the semantic category download URL to navigate the user's web browser to the homepage for the semantic category type applied to the string. For example, suppose new actions have been defined for the semantic category type "person name". If so, then new actions will be downloaded to the user's computer after selecting "Check for new actions..." **425**. "Check for new actions..." **425** will be grayed out if a download URL is unavailable for the semantic category.

If selected, the "Remove this semantic category" item **430** deletes the semantic

category label from the string. If selected, the "Semantic categories" item **435** navigates the user to the semantic categories tab of the autocorrect dialog.

It should be understood that the application program module sends a request to the action DLL to determine which actions are shown with each semantic category type.

Actions Performed in Association with Semantic categories

There are a number of functions that users perform on typed data that preferred word processor program module **37** and semantic categories will make easier. The functions fall into three primary categories:

1) interacting with personal information manager contacts, tasks, meetings, and mail;
2) interacting with properties on the World Wide Web or a corporate intranet; and
3) interacting with other applications on the client machine.

A single string may be associated with multiple semantic categories. Every semantic category has a type label with one or more action plug-ins defined for the type label. For example, the "Address" type label may have the "Open in Mappoint", "Find with Expedia Maps" and "Add to my Address Book" actions associated with it and each of these actions may have a different action plug-in to execute the action.

The actions assigned to type labels also depend on the computer that the application program module is running on. Thus, if a computer has three actions registered for the type label "Address", then all strings with an "Address" type label will be assigned to three actions. However, if one of these semantic categories is sent to a computer which has only two actions registered for the "Address" type label, then the user will only be exposed to two actions for this semantic category.

Nesting of Semantic categories

In an embodiment of the present invention, semantic categories may be nested inside each other. For example, the string "George Washington" may include a semantic category with type label "Person Name" for the span "George Washington State" and a

semantic category with type label "State" for the span "Washington". Moreover, two semantic categories may cover exactly the same span. For example, the string "George Washington" may include a semantic category with type label "Person Name" and a semantic category with type label "President".

Because the preferred application program module **37** will support labeling a single string with multiple type labels (e.g. Bob Smith could be a semantic category labeled as a "Person Name" and labeled as a "Microsoft employee"), the preferred application program module **37** will use cascade menus on the dropdown menu if multiple semantic category types are assigned.

For example, the cascade menu may include a list of the type labels included in the recognized string. This list may include a type label "Person Name" and a type label "Microsoft employee".

It should be understood that a cascade menu may be used to allow the user to select which type label the user is interested in and to further select an action after selecting the type label.

In-document User Interface to Indicate Semantic categories

As described above with reference to Fig. 4, the application program module may include the option to display an in-document user interface to indicate the location of semantic categories. This in-document user interface may use a colored indication to indicate the location of a semantic category, such as the brackets **415** in Fig. 4. The in-document user interface will also be able to show nesting of semantic categories. For example, if Michael Jordan is labeled as a semantic category with type label "Person Name", Michael is a semantic category with type label "First Name" and Jordan is a semantic category with type label "Last Name", the document may look like this with the brackets indicating semantic categories:

[[Michael][Jordan]]

Of course, the in-document user interface may be any sort of indication. For example, in the "EXCEL" spreadsheet application program, the interface comprises a

triangle in the lower right hand portion of a cell to indicate that one or more semantic categories are present in the cell.

Although the present invention has been described as implemented in a word processing program module, it should be understood that the present invention may be implemented in other program modules, including, but not limited to, HTML authoring programs and programs such as the "POWERPOINT"® presentation graphics programand the "OFFICE" program module, both marketed by Microsoft Corporation of Redmond, Washington.

As described above, the semantic category may also include metadata returned by the recognizer plug-ins. For example, a recognizer plug-in that recognizes the titles of books may return as metadata an ISDN book number when it recognizes the title of a book. The ISDN book number metadata may then be used to provide actions. Metadata may also be used to disambiguate for actions and searches. For example, suppose a recognizer DLL is linked to a corporate employee database to recognize names. When the recognizer DLL recognizes "Bob Smith", it may store "employeeID=12345" as metadata in the background. Then, when an action is fired, the text in question will be known to reference Bob Smith, employee no. 12345 rather than Bob Smith, employee no. 45678. Also, the metadata may allow searches to be performed independent of the actual text in a document. So, a search may be conducted on "Robert Smith" by looking for employee 12345 in the employee databases and by performing a search on the metadata for employee number 12345 to find documents with "Bob Smith" in them. There are also numerous other functions for metadata. For instance, DHTML could be inserted so special tricks may be performed within a web browser. Additionally, data used by other actions may be inserted such as someone's e-mail address that could be used by the send-mail-to action, a normalized version of the date could be stored to easily interact with a personal information manager, etc.

Providing Electronic Commerce Actions

The present invention uses semantic categories to provide electronic commerce

related functions.

One key aspect of recognizer plug-ins is that, in addition to applying category information to a string, they can also associate arbitrary metadata with that string. For example, the recognizer plug-in for recognizing person names might also embed the person's employee id number in the metadata of the semantic category. The benefit of embedding additional metadata is that the pool of metadata is available to the action plug-ins. Thus, for example, an action plug-in for sending someone an e-mail might most easily be written if the employee id number were embedded in the semantic category as metadata, along with the name, assuming that employee e-mail addresses were derived from employee numbers.

There are numerous uses of semantic categories and numerous uses of the mechanism for embedding arbitrary metadata along with a recognized string of a semantic category. One use of particular importance is using the metadata in conjunction with electronic commerce, or e-commerce. Semantic categories are particularly useful because they can be embedded in web pages and in HTML mail messages, two of the main avenues of e-commerce in use today. Third parties will find semantic categories particularly useful because of the ease of developing recognizer plug-ins and action plug-ins.

It should be understood from the foregoing description that semantic categories are a technology that allow for a new navigation model on the World Wide Web and in applications. A hyperlink allows for a static, one-to-one navigation model and is typically not applied automatically to document content. In contrast, semantic categories allow for a one-to-many navigation model (a word or phrase with a single type label applied can offer multiple actions). For example, using the present invention, a user may have the option to buy a book, look for books by the same author or give the book as a gift. Moreover, using the present invention, a single string may have multiple type labels applied, so "Grapes of Wrath" could be labeled as both a book and a DVD and the user would be presented with options related to both of these type labels, such as buy the

book, buy the DVD, etc.

Semantic categories also improve on the hyperlink by being dynamic. Actions can perform arbitrary computation in addition to performing navigation within a web browser. For example, an action could communicate to a web server the identity of the user in the form of their customer number without using cookies. Or, an action could communicate with a web service to notify an e-commerce retailer that the user was directed to their site via semantic categories and, going one step further, could even communicate the type of semantic category from which the navigation began as well as the contents of the semantic category, which could include information about who authored the document containing the semantic category.

Another example of semantic categories being dynamic is that they can contain client-side logic to replace server-side logic and/or redirection services. For example, an e-commerce provider might want to provide the user with the contents of their web site in the appropriate language. The e-commerce provider could do this by showing the user an initial web page and asking the user to choose the appropriate language, or the provider could show the user a web page and run some script to check the language of the user's web browser. However, both of these approaches require the use of an interim web page. Using semantic categories, this information can be passed to the web site at run time, which is an improvement over the static nature of Universal Resource Locators (URLs).

Furthermore, semantic categories are an improvement over hyperlinks in that they are applied automatically. Some applications automatically recognize hyperlinks when they are typed and turn them into links. However, semantic categories allow developers to specify specific strings that should be labeled when the user types them (not necessarily just strings that are hyperlinks). For example, an online CD retailer might want all names of bands labeled in web pages, documents and mail messages when the user types them. After labeling the names of bands, the retailer could direct the user to a web site with a single click rather than forcing the user to 1) remember the name of their company, 2) remember the URL of their web site, 3) enter that URL in the browser and

4) search for the band on the retailer's website.

Thus, as should be understood from the foregoing description, by virtue of the dynamic nature of semantic categories a number of e-commerce scenarios are enabled and a number of other scenarios become much simpler.

E-commerce Uses of Semantic Categories

One e-commerce use of semantic categories is labeling the names of products using recognizer plug-ins and offering users the opportunity to buy those products from a particular e-commerce company via an action plug-in. For example, it would be useful to a CD retailer to recognize the names of albums and offer to sell those albums to users.

Another use of semantic categories is as a mechanism to identify customers who visit a web site. Due to security concerns, some users turn off cookies. As a result, it can be difficult to track the identity of users who visit web sites. Subject to privacy concerns, users who navigate to a particular web site via semantic category actions (even with cookies turned off) may have their identity transmitted to the web site by the action plug-in that allowed for navigation to that website.

Still another use of semantic categories is for electronic commerce coupon programs. There are a number of ways electronic coupons can be issued with semantic categories. In conjunction with identifying the customer as described above, discounts and promotional offers can be made based on the identity of the shopper. Electronic coupons can also be embedded in the semantic category by the recognizer plug-in. For example, when recognizing the album title "What's the story morning glory", a recognizer plug-in could embed an electronic coupon for that album in the semantic category. Viewing the HTML source of the document containing this album name might reveal something like this:

<st1:album discount= "10%" couponID= "123456789">What's the Story Morning Glory</st:album>

The st1:album designates "What's the Story Morning Glory" as an album. The attribute value pair discount= "10%" specifies that the 'coupon' is good for 10% off. CouponID=

"123456789" identifies the coupon and could be drawn from a very sparse space of coupon numbers so that users couldn't easily forge their own coupons, just as it is difficult to forge credit card numbers because few of the possible numbers are valid. The action plug-in may offer the user the option to "Buy this album". When the user executes that action, the user's web browser would be taken to the coupon issuer's web site to purchase the album and the discount and couponID attribute/value pairs would be passed to the web site. The web site would interpret these attribute/value pairs and apply the discount after checking the validity of the coupon. It could also log the use of the coupon so that it could only be used once. There are a number of other attribute/value pairs that the recognizer plug-in could specify which in turn could be interpreted by the seller's web site, such as a begin and end date for the coupon, special terms of the coupon (credit card orders only, only valid when purchasing more than $50 worth of goods).

Another use of semantic categories is for electronic coupons that depend on user behavior. For example, retailers can incorporate into action plug-ins and recognizer plug-ins code that tracks user behavior and issues coupons based on that behavior. For example, users who frequently mention names of products in their documents might be deemed by a seller as being a "recommender" of their products and could as a result be offered special incentives. This may work as follows: a seller's recognizer plug-in could track the number of different products that it recognized on the user's machine and offer discounts as a result of frequent labeling. Alternatively, a seller's action plug-in could track the number of times a user navigated to the seller's web site using an action and reward them for frequently visiting. One of the advantages of these approaches is that it's completely client-side and tracks different user behavior than is possible by simply collecting statistics about web site visitation.

Another use of semantic categories is for electronic coupons that depend on buying behavior. More than simply tracking web site usage locally, retailers can track purchases locally as well and offer incentives to infrequent or lapsed buyers if they choose. For example, an e-commerce company may build into their action DLL a

tracking component so that the e-commerce company does not have to expend server side processing to determine whether a customer has made a recent purchase. Using the action DLL, if an action has not been used to make a purchase for a predetermined period of time, such as 30 days, then a new action may be added to the menu to induce the buyer to make a purchase. For example, a buy one get one free action may be added if a user has not made a purchase recently.

Still another use of semantic categories is for buyer reward programs. A number of companies have affiliate or buyer reward programs in which people who recommend products to friends and acquaintances either by mentioning them on their web site or in e-mail are rewarded for their recommendation. Use of such programs requires effort on the recommender's part because it typically involves inserting hyperlinks manually into documents and mail messages. With semantic categories, recognizer plug-ins can automatically embed affiliate number identification when a product is labeled and the action plug-in can convey that information to the seller's web site. For example, if a user recommends an album to a friend, the underlying HTML might look like this: <st1:album affiliateID= "123456789">What's the Story Morning Glory</st:album> When the user's friend received the document or message containing the recommendation, he may use an action plug-in to make the purchase. The action would convey the affiliateID to the web site and the web site would give the user "credit" for making this recommendation.

There are more sophisticated ways to use recommendation information as well. Typically recommenders are rewarded based solely on the number of sales that result from their recommendations, but recognizer plug-ins could be instrumented in order to provide sellers with statistics about how many recommendations their affiliates make. In some cases it might be beneficial to reward people based on metrics other than the total number of sales, such as their conversion rate, the number of recommendations they make, the variety of recommendations they make, etc. For example, the recognizer DLL tracks how many times it labels something as a certain type label, e.g. compact disk. The

recognizer DLL also tracks which uniques IDs were applied. When the recognizer DLL boots up each day, it converses with the e-commerce company's web server to obtain sales information for each of the unique IDs. The recognizer DLL may then use an analysis of this sales information to provide incentives, such as: providing discounts to people who convert ten percent or more of their recommendations into sales, providing discounts to people with a high number of purchases per recommendation (these are efficient recommenders and are most likely those who write things that are published), providing discounts to those recommenders with the most total recommendations, etc.

Semantic categories may also be used to simplify the process of buying multiple items from an e-commerce site that can be particularly tedious because it usually involves repeated searching. Action plug-ins may be given access to the contents of the document and can in some applications easily identify other semantic categories besides the one with which they are associated. As a result, "buy all" actions are possible. For example, a user may send e-mail to a friend saying that he'd recently purchased three compact discs and listing each of them in the e-mail. The friend may choose to purchase them all based on the user's recommendation. Action plug-ins make these purchases simple by providing an action to buy all items of a particular type mentioned in a document.

Semantic categories may also be used to implement coupon and recommendation e-mail/web pages. Many e-commerce companies distribute electronic coupons and recommendations, such as new or featured product announcements, via e-mail. Typically these e-mail messages include hyperlinks that allow users to purchase goods. Semantic category type labels (whether manually or automatically applied) and actions can be used instead of hyperlinks and convey many advantages such as: multiple actions for a single piece of data, they can be dynamic, etc. For example, a CD retailer might send an e-mail message about a featured band and the band name could be tagged with a type label in the e-mail message. Rather than simply linking the band name to the band's discography with a hyperlink, the retailer could deploy a set of actions that allow the user to look up the discography, find related bands, purchase their entire catalog of CDs, retrieve

information about upcoming concerts, etc.

Semantic categories may also be used to label words and phrases associated with products and provide navigation to a retailer's website. In addition to product names themselves, there are a number of other sorts of information that may be pertinent to selling goods. So, names of manufacturers, names of authors, names of bands, and names of categories of merchandise can all be labeled by recognizer plug-ins. Once labeled as semantic categories, they can be used by action plug-ins to provide users with e-commerce opportunities. For example, a recognizer plug-in might label variants of the word "computer" and offer to take users to a computer retailer's website.

In another embodiment, semantic categories may be used to label words associated with shopping in a user's documents and provide actions to navigate to shopping web sites. Thus, in addition to labeling product names and other data particular to different types of products, commerce-related words could also be labeled. For example, it might be beneficial to label words like "buy" and "sell" and offer actions to take a user to a retailer's general purpose shopping websites.

In yet another embodiment, semantic categories may be used to mine a user's recognition results to determine the user's preferences. Many electronic commerce companies currently use techniques such as collaborative filtering to determine user preferences and make recommendations about future purchases. For example, a CD retailer could collect data about a customer's purchasing habits and compare that data to purchasing data aggregated across multiple customers to recommend music that other people with statistically similar tastes (based solely on purchasing habits from that retailer) also purchased. A problem with this approach is that it erroneously makes a closed-world assumption---namely that the customer has bought all of their music from the retailer and as a result their purchase list represents their taste in music. However, this is rarely the case. Many users buy from multiple sources, including physical stores. A recognizer plug-in on the user's machine could, however, track the names of all bands the user typed or read about in order to get a richer portrait of the customer's preferences.

In addition, that recognizer plug-in could scrutinize the customer's media player play list as an additional source of data. Similar examples apply to other categories of purchase such as: books, electronics, clothes, etc.

Semantic categories may be used to implement recommendation systems based on multiple sorts of data on user's machine. In addition to being able to build simple recommendation systems that perform collaborative filtering over single types of goods (e.g. books or CDs), semantic categories present the opportunity for retailers who sell broad varieties of goods to develop client-side profiles that represent the user's interests in a wide variety of products. This information can be intermittently transmitted to the seller in order to expand its database of customer profiles and in order to improve the collaborative filtering process.

Semantic categories may also be used to implement recommendation systems based on information other than product names and purchasing history. Shopping software on the client machine can access information about the software and computer the user possesses and could make recommendations based on that information as well. For example, knowing that the user had an ink jet printer of a particular make and model may allow a seller of printer supplies to offer useful discounts on toner cartridges or photo-quality ink jet printer paper.

In other embodiments, semantic categories may be used to implement client-side shopping application software. Current e-commerce initiatives revolve around directing users to websites. Although this approach works fairly well, it has the limitation of using HTML (or an animation package such as Flash or Director) to provide the user interface. With semantic categories, it's possible to build actions that allow users to shop and buy using the full richness of the operating system desktop user interface standards. Semantic categories provide a natural entry point for such a shopping application in applications, e-mail messages and on the Web.

Semantic categories may be used to implement client-side shopping applications designed to minimize server load. Numerous companies have built comparison shopping

services (e.g. MySimon.com). There are a number of technical difficulties in building and maintaining these services including: 1) many shopping sites block spider and robot programs from harvesting information from the sites and, as a result, block comparison shopping services from retrieving the data they need; and 2) there is a large demand on server software due to the need to remain up-to-date in a marketplace with ever–changing prices because of the need to query shopping sites frequently to avoid making recommendations with out-of-date information. Building a client-side shopping application using semantic categories has the advantage that the shopping application is run on the client's machine and that machine bears the brunt of the burden for communicating with other web sites. Thus, the model is changed from one of caching prices on the server and delivering information about the best deals to the user when they browse the comparison shopping service's web site to having the client-side shopping application poll a number of web sites when the user makes a purchase or price request. This use of semantic categories essentially moves the comparison service from a client-server paradigm to a peer-to-peer paradigm. The client application could host advertising or the user could pay a monthly fee or a transaction fee for use of the client-side comparison pricing service.

In still another embodiment, semantic categories may be used to store product identity information. There are often numerous similarly named products, including different editions of books, products that differ only in terms of model year, etc. However, often times there is a product identification code that distinguishes among them. For example, books have ISBN numbers that distinguish between paperback and hardback versions. It may be useful for applications to store product information in the semantic category when a string is labeled as such to facilitate e-commerce. In some cases, this labeling may be trivial if there is one version of a particular product. However, in other cases there may be many versions of a particular product. In these cases, computation may be performed on the user's machine based on attributes such as the language of the text, the location of the user's machine, the current date, information

about the user's buying preferences, etc. For example, the book "Harry Potter and the Goblet of Fire" was released simultaneously in the U.S., the U.K. and elsewhere. U.S. users typically want to purchase U.S. editions of books so the semantic category might contain the ISBN for the U.S. edition. However, determining that the user is in the U.S. requires that the recognizer plug-in evaluate various information on the user's machine, possibly including the version of the "OFFICE" suite (or other software) installed. This evaluation may include determining the locale of the user's machine, checking the locale as set in the operating system, checking the current time zone, checking the language used by the user interface, checking the user's profile with the bookseller, etc.

Semantic categories may also be used to attach unique identities to instances of product names. As a retailer, it might be useful to know which instances of particular mentions of products are used to initiate sales of those products. This identification goes one step further than affiliate programs, which have the identity of the recommender as the most granular level of detail. This identification may work as follows: every time a recognizer plug-in labels a particular string with a particular type label, the recognizer plug-in may insert an attribute/value pair into the semantic category as metadata. This attribute/value pair is essentially a globally unique identifier (GUID)—a 128 bit number that uniquely identifies this recognition event. Actions initiated from the labeled semantic category may be built in such a way that the GUID is passed to the web site involved in the e-commerce transaction when the action is requested. This allows the retailer to track which instances of a recommendation are most successful, as well as how many recommendations (or, more specifically, mentions) of a particular product are made by a particular customer. It also allows purchasing behavior and network effects to be tracked. For example, user A might recommend a book (with GUID x) to a user B in an e-mail message. User B might forward the recommendation (or copy and paste, which would preserve the GUID) to two friends, who in turn might forward it to two friends each and so on. Current affiliate programs typically give the credit for the recommendation to the person who made the most recent recommendation and have no

mechanism for tracking this sort of network effect, because each recommendation is identical. Placing GUIDs in semantic categories would allow these sorts of effects to be studied and rewarded so that particular individuals could be targeted with promotional information, coupons or other devices that encourage them to purchase and subsequently recommend what they purchase.

In yet another embodiment, semantic categories may be used to retrieve document contents during a transaction initiated from an action. Action plug-ins have the opportunity to gather information about the document from which the action was initiated. This data can then be used in whatever programmatic way the author of the action plug-in chooses. In some cases, subject to privacy constraints, it might be useful to transmit this information to the author of the action plug-in. This information may enable the retailer to examine data about recommendations in order to determine whether certain forms of recommendations are more powerful or generate additional sales. This information could be used to alter the wording of the retailer's promotional offers, coupons and web site content.

Although the present invention has been described above as implemented in a preferred application program module, it will be understood that alternative embodiments will become apparent to those skilled in the art to which the present invention pertains without departing from its spirit and scope. Accordingly, the scope of the present invention is defined by the appended claims rather than the foregoing description.